RULE RELAXATION AND SUBSET OPTIMIZATION SYSTEM

By Inventors:

Michael Neal
Krishna Venkatraman
Rob Parkin
Suzanne Valentine
Phil Delurgio
Hau Lee

CROSS-REFERENCE TO RELATED APPLICATIONS

This application relates to co-pending and concurrently filed application No. _____ (Attorney Docket No. DT.0106) filed November 30, 2001, entitled "Selective Merchandise Price Optimization Mechanism", by Michael Neal, Krishna Venkatraman, Rob Parkin, Suzanne Valentine, Phil Delurgio, Hau Lee, and John Close, which is incorporated by reference herein for all purposes.

BACKGROUND OF THE INVENTION

The present invention relates to providing optimized pricing for a subset of a plurality of products for a plurality of stores.

In businesses, prices of various products must be set. Such prices may be set with the goal of maximizing profit or demand or for a variety of other objectives. Profit is the difference between total revenue and costs. Total sales revenue is a function of demand and price, where demand is a function of price. Demand may also depend on the day of the week, the time of the

year, the price of related products, the location of a store, and various other factors. As a result, the function for forecasting demand may be very complex. Costs may be fixed or variable and may be dependent on demand. As a result, the function for forecasting costs may be very complex. For a chain of stores with tens of thousands of different products, forecasting costs and determining a function for forecasting demand are difficult. The enormous amounts of data that must be processed for such determinations are too cumbersome even when done by computer.

It is desirable to provide an efficient process and methodology for determining the prices of individual products such that profit (or whatever alternative objective) is optimized.

During optimization, it may be found that one of the plurality of rules is infeasible. Such a finding of infeasibility may stop the optimization process. It would be desirable to provide a method of handling optimization when a rule is found to be infeasible.

It is desirable to update prices as new information about the products and competitive environment is received. Examples of this type of information would include product cost updates, product addition or discontinuance, and new competitive price or base price data. Ideally, when a user receives new information, they would run a secondary optimization over a whole product category. This would allow the optimization to fully capture all of the complementary and substitution effects in selecting the new set of

optimal prices. However, if the cost of changing prices is not negligible, optimization and changing prices of all products in a whole product category may not be desirable.

SUMMARY OF THE INVENTION

To achieve the foregoing and other objects and in accordance with the purpose of the present invention, a method for computing a preferred set of prices for a subset of a plurality of products is provided. Generally, initial prices for a plurality of products are stored. A subset of products of the plurality of products is designated, where the number of products in the subset of products is less than the number of products in the plurality of products. Prices for the subset of products are optimized, while maintaining the initial prices of products of the plurality of products that are not in the subset of products.

In another embodiment of the invention, an apparatus for computing a preferred set of prices for a subset of a plurality of products, comprising computer readable media, is provided. Generally, the computer readable media comprises computer readable code for storing initial prices for a plurality of products, computer readable code for designating a subset of products of the plurality of products, wherein the number of products in the subset of products is less than the number of products in the plurality of products, and computer readable code for optimizing prices for the subset of

product, while maintaining the initial prices of products of the plurality of products that are not in the subset of products.

In another embodiment of the invention, a method for setting prices for a subset of products of a plurality of products is provided. Optimized prices for a product category are received. Every item in the product category is priced according to the received optimized prices. New data is then provided. New prices for the subset of products of the product category are then received, where the subset is smaller than the product category, and where the received new prices are generated by storing initial prices for a plurality of products, designating a subset of products of the plurality of products, and where the number of products in the subset of products is less than the number of products in the plurality of products, and optimizing prices for products in the subset of products, while freezing the initial prices of products of the plurality of products in the product category that are not in the subset of products. Prices for the subset of products are then set according to the received new prices.

In another embodiment of the invention, computer data signal embodied in a carrier wave and representing sequences of instructions which when executed by a processor, causes the processor to compute a preferred set of prices for a subset of a plurality of products is provided. Initial prices for a plurality of products are stored. A subset of products of the plurality of products is designated, where the number of products in the subset of products is less than the number of products in the plurality of products. Prices for products in the subset of products are optimized, while maintaining the initial prices of products of the plurality of products that are not in the subset of products.

In another embodiment of the invention a price database is provided. Initial prices for a plurality of products are stored. A subset of products of the plurality of products is designated, where the number of products in the subset of products is less than the number of products in the plurality of products. Prices for products in the subset of products are optimized, while maintaining the initial prices of products of the plurality of products that are not in the subset of products.

In another embodiment of the invention, a method for obtaining optimized price data on a client system is provided. Sales data is sent to a server system for a plurality of products. Optimization preferences are selected. The optimization preferences are sent to the server system. Optimization prices for all of the plurality of products are received from the server system. Additional sales data is sent to the server system. A subset constraint is selected. The subset constraint is sent to the server system. A new set of optimization prices for a subset of the plurality of products is received, where the number of products in the subset is less than the number of the plurality of products.

These and other features of the present invention will be described in more detail below in the detailed description of the invention and in conjunction with the following figures.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

FIG. 1 is a high level schematic view of a price optimizing system.

FIG. 2 is high level flow chart of a price optimization process.

FIG. 3 is a more detailed schematic view of an econometric engine.

FIG. 4 is a more detailed schematic view of an optimization engine and support tool.

FIG. 5 is a block diagram to illustrate some of the transaction costs that occur in retail businesses of a chain of stores.

FIG. 6 is a flow chart of a preferred embodiment of a rule relaxation process.

FIG. 7 is an overall flow chart of a process that uses subset optimization.

FIG. 8 is a more detailed flow chart of an embodiment of a step of optimizing product category within subset limits.

FIG.'S 9A and 9B illustrate a computer system, which forms part of a network and is suitable for implementing embodiments of the present invention.

FIG. 10 is a schematic illustration of an embodiment of the invention that functions over a network.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention will now be described in detail with reference to a few preferred embodiments thereof as illustrated in the accompanying drawings. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without some or all of these specific details. In other instances, well known process steps and/or structures have not been described in detail in order to not unnecessarily obscure the present invention.

To facilitate discussion, FIG. 1 is a schematic view of a price optimizing system 100, which may be used in rule relaxation and subset optimization. The price optimizing system 100 comprises an econometric engine 104, a financial model engine 108, an optimization engine 112, and a support tool 116. The econometric engine 104 is connected to the optimization engine 112, so that the output of the econometric engine 104 is an input of the optimization engine 112. The financial model engine 108 is connected to the optimization engine 112, so that the output of the financial model engine 108 is an input of the optimization engine 112. The optimization engine 112 is connected to the support tool 116 so that output of the optimization engine 112 is provided as input to the support tool 116 and output from the support tool 116 may be provided as input to the optimization

engine 112. The econometric engine 104 may also exchange data with the financial model engine 108.

To facilitate understanding, FIG. 7 is an overall flow chart of a process that uses subset optimization 700. First, a product category is optimized (step 701). A demand group is defined as a set of products that are substitutes or near substitutes for each other. A product can belong to only one demand group. A product category consists of one or more demand groups. FIG. 2 is a more detailed flow chart of a preferred embodiment of a process that utilizes the price optimizing system 100 to optimize prices for a product category (step 701). Data 120 is provided from the store computer systems 124 to the econometric engine 104 (step 204). Generally, the data 120 provided to the econometric engine 104 may be point-of-sale information, product information, and store information. The econometric engine 104 processes the data 120 to provide demand coefficients 128 (step 208) for a set of algebraic equations that may be used to estimate demand (volume sold) given certain marketing conditions (i.e. a particular store in the chain), including a price point. The demand coefficients 128 are provided to the optimization engine 112. Additional processed data from the econometric engine 104 may also be provided to the optimization engine 112.

The financial model engine 108 may receive data 132 from the store computer systems 124 (step 216) and processed data from the econometric engine 104. The data 132 received from the stores is generally cost related

data, such as average store labor rates, average distribution center labor rates, cost of capital, the average time it takes a cashier to scan an item (or unit) of product, how long it takes to stock a received unit of product and fixed cost data. The financial model engine 108 may process the data to provide a variable cost and fixed cost for each unit of product in a store. The processing by the econometric engine 104 and the processing by the financial model engine 108 may be done in parallel. Cost data 136 is provided from the financial model engine 108 to the optimization engine 112 (step 224).

The optimization engine 112 utilizes the demand coefficients 128 to create a demand equation. The optimization engine is able to forecast demand and cost for a set of prices and promotion conditions such as store displays and ads to calculate net profit. The stores 124 may use the support tool 116 to provide optimization rules to the optimization engine 112 (step 228). The optimization engine 112 may use the demand equation, the variable and fixed costs, and the rules to compute an optimal set of prices that meet the rules (step 232). For example, if a rule specifies the maximization of profit, the optimization engine would find a set of prices that cause the largest difference between the total sales revenue and the total cost of all products being measured.

If a rule providing a promotion of one of the products by specifying a discounted price is provided, the optimization engine may provide a set of prices that allow for the promotion of the one product and the maximization of

profit under that condition. The rules normally include an objective, such as optimizing profit or optimizing volume of sales of a product and constraints such as a limit in the variation of prices

When profit is maximized, it may be maximized for a sum of all measured products. Such a maximization, may not maximize profit for each individual product, but may instead have an ultimate objective of maximizing total profit.

The optimal (preferred) set of prices may be sent from the optimization engine 112 to the support tool 116 so that the stores 124 may use the user interface of the support tool 116 to obtain the optimal set of prices. Other methods may be used to provide the optimal set of prices to the stores 124. The price of the products in the stores 124 is set to the optimal set of prices (step 236), so that a maximization of profit or another objective is achieved.

Each component of the price optimizing system 100 will be discussed separately in more detail below.

FIG. 3 is a more detailed view of the econometric engine 104. The econometric engine comprises an imputed variable generator 304 and a coefficient estimator 308. The data 120 from the stores 124 is provided to the imputed variable generator 304. The data 120 may be raw data generated from cash register data, which may be generated by scanners used at the cash registers. The raw data may be processed to create imputed variables. The

imputed variables may be used to estimate parameters for a demand equation D(x), which may be used to estimate the amount of sales using variables such as price.

In a preferred embodiment of the invention, sales for a demand group (S) is calculated and a market share (F) for a particular product is calculated, so that demand (D) for a particular product is estimated by D=S•F. A demand group is defined as a collection of highly substitutable products. More preferably the creation of the demand model relies on a mixed-model framework, simultaneously utilizing information across all stores and products in a client category, where a category is defined as a collection of substitutable or complementary products. The mixed model methodology is also referred to as "Bayesian Shrinkage" Modeling, because by combining data from various stores and/or products, one can "shrink" individual parameter estimates towards the average estimate, dampening the extreme values that would result if traditional regression were used.

In developing product-level volume models for each store within a chain, one may be presented with a wide range of historical data in terms of modeling sufficiency. For some stores and/or products, the history will be quite rich, with many price changes and a variety of promotion patterns. For other stores and/or products, the history will be quite sparse, with very few price changes and little promotional activity. To maximize the stability of estimated model parameters, one might consider developing a single

regression model across stores and/or products. This model might have stable parameter estimates; however, it would not fully leverage the store and/or product variability, and the resulting model would likely not predict well for a particular store and/or product. On the other hand, one might consider developing individual regression models for each store and/or product, to utilize the specific information contained in the data history. While these models might fit and predict well for the stores and/or products with substantial price variability, models would not be estimable for stores and/or products without a rich data history.

A mixed-model framework addresses the need for both highly predictive models and the existence of an estimable model for each store and product. In a mixed-effect model, information (in the form of data history) is leveraged across all stores and products, and a single cohesive model is built. Stores and products with little or no information in their data history default to an "average" (fixed-effect) model. Likewise, stores and products with a wealth of information in their data history will end up with unique parameter estimates tailored to their response pattern, via estimation of non-zero store and/or product-specific adjustment factors (random effects) which are added to the fixed-effect portion of the model.

The financial model engine 108 receives data 132 from the stores 124 and may receive imputed variables and data from the econometric engine 104 to calculate fixed and variable costs for the sale of each item. To facilitate

understanding, FIG. 5 is a block diagram to illustrate some of the transaction costs that occur in retail businesses of a chain of stores. The chain of stores may have a headquarters 504, distribution centers 508, and stores 512. The headquarters 504 may place an order 516 to a manufacturer 520 for goods supplied by the manufacturer 520, which generates an order placement cost. The manufacturer 520 may ship the goods to one of the distribution centers 508. The receiving of the goods by the distribution center 508 generates a receiving cost 524, a cost for stocking the goods 528, and a cost for shipping the goods 532 to one of the stores 512. The store 512 receives the goods from one of the distribution centers 508 or from the manufacturer 520, which generates a receiving cost 536 and a cost for stocking the goods 540. When a customer purchases the item, the stores 512 incur a check-out cost 544. With the large number of retail chains, different purchasing and delivery processes may be used. Even within a single chain, different manufacturers may provide different invoicing and delivery procedures and costing system, which may be different than the processes illustrated in FIG. 5.

The financial model engine 108 should be flexible enough to provide a cost model for these different procedures. These different costs may have variable cost components where the cost of an item is a function of the amount of sales of the item and fixed cost components where the cost of an item is not a function of the amount of sales of the item. The financial model engine 108 may use these fixed and variable costs to determine $C_{s,i,k,t}$, where $C_{s,i,k,t}$, is a cost for a particular product (k) given a store (s), demand group (i), and a day

(t). The financial model engine 108 may use industry data to provide standard estimates. For example, instead of measuring how long it takes to stock a box of an item, an industry data may be used to estimate this time. The standard estimates helps to reduce the amount of data that must be collected. In a preferred embodiment of the invention, the stores may only need to supply labor costs of the stores and distribution centers, cost of capital, size of an item, and number of items in a case to allow a cost modeling. By using these estimations, costs may be more easily calculated on a store level, instead of being averaged over all of the stores. This is because if large amounts of data were measured by hand, measuring such data for each store would be difficult. The tailoring of costs per store allows the maximization of profits for each store.

FIG. 4 is a more detailed schematic view of the optimization engine 112 and the support tool 116. The optimization engine 112 comprises a rule tool 404 and a price calculator 408. The support tool 116 comprises a rule editor 412 and an output display 416.

In operation, the client (stores 124) may access the rule editor 412 of the support tool 116 and provides client defined rule parameters (step 228). If a client does not set a parameter for a particular rule, a default value is used. Some of the rule parameters set by the client may be constraints to the overall weighted price advance or decline, branding price rules, size pricing rules, unit pricing rules, line pricing rules, and cluster pricing rules. These rules will be

discussed in more detail regarding a preferred embodiment of the invention. The client defined parameters for these rules are provided to the rule tool 404 of the optimization engine 112 from the rule editor 412 of the support tool 116. Within the rule tool 404, there may be other rules, which are not client defined, such as a group sales equation rule. The rule parameters are outputted from the rule tool 404 to the price calculator 408. The demand coefficients 128 and cost data 136 are also inputted into the price calculator 408. The client may also provide to the price calculator 408 through the support tool 116 a desired optimization scenario rules. Some examples of scenarios may be to optimize prices to provide the optimum profit, set one promotional price and the optimization of all remaining prices to optimize profit, or optimized prices to provide a specified volume of sales for a designated product and to optimize price. The price calculator 408 then calculates optimized prices. The price calculator 408 outputs the optimized prices to the output display 416 of the support tool 116, which allows the stores 124 to receive the optimized pricing (step 232).

The system can optimize for profit, revenue, and sales. For each type of objective, users can impose constraints on the values of the other two. For instance a user can optimize profit while ensuring that total sales and revenue do not fall below their original values, or he/she can optimize sales while requiring profit to increase by 1%.

A preferred embodiment of the optimization engine uses the group sales equation S(x) and the market share equation F(x), previously described, to predict group sales and product market share, respectively. These two are then combined to predict product sales at the store level D(x)=S(x)•F(x).

If the objective is to maximize profit then the following equation sum is maximized:

$$\sum_{t\in G}\sum_{k\in Dem_i}\hat{D}_{s,t,k,t}(P_{s,t,k,t}-C_{s,t,k,t})$$
$$=\sum_{t\in G}\sum_{k\in Dem_i}\hat{F}_{s,t,k,t}\hat{S}_{s,t,t}(P_{s,t,k,t}-C_{s,t,k,t})$$

The above mentioned rules may be constraints forming boundaries. These constraints are rules that provide limitations to price or other changes. In a preferred embodiment such constraints may be used as follows:

1. Group price advance or decline.

Since demand groups are made up of like or substitutable products, managers often wish to constrain the overall weighted price advance or decline in price for them, where the weights are the market shares of the products that constitute the demand groups. The constraints are:

$$\forall s,i\ ,\ PMIN_{s,i}\le\sum_{k}F_{s,t,k}P_{s,t,k}\le PMAX_{s,i}$$

2. Brand Pricing Rules:

Products are described and categorized by attributes such as brand, size and flavor. Therefore, each product may be associated with a set of attributes and values. These attributes are useful to us for several reasons. The attributes may be used in the regression modeling in order to create a parsimonious set of regression coefficients. They may also be used in setting rules. For instance, category managers might wish to offer store brands at lower prices compared to the competing national brands. They might also wish to constrain some product brands to be less expensive than others, either for strategic considerations or to meet their contractual obligations. Specifically, a manager can create a set $\text{Brand}_{s,i} \equiv \left\{\left(p_{s,i,l_1}, p_{s,i,k_2}\right): p_{s,i,k_1} \text{ must cost less than } p_{s,i,k_2}\right\}$, which leads to the following constraints:

$$\forall s, i \text{ and } \left(p_{s,t,k_1}, p_{s,t,k_2}\right) \in Brand_{s,t}\ p_{s,t,k_1} \le p_{s,t,k_2}$$

3. Size Pricing Rules:

Managers might also wish to create rules that relate the price of one product versus another based on their sizes. For instance, they might wish for products belonging to the same brand and sharing other attributes, but with different sizes to be priced such that the larger sized product costs less per equivalent unit of measure than a smaller one. Then

$\text{cSize}_{s,i} \equiv \{(p_{s,i,l_1}, p_{s,i,k_2}): p_{s,i,k_1}$ must cost less than $p_{s,i,k_2}\}$, and create a set , which leads to the following constraints:

$$\forall s,i \text{ and } (p_{s,t,k_1}, p_{s,t,k_2}) \in Size_{s,i}\ p_{s,t,k_1} \le p_{s,t,k_2}$$

4. Unit Pricing Rules:

Continuing in the same vein, managers might wish to ensure that two products that are identical in every respect but size should be priced such that the larger product costs more than the smaller one. This rule is closely related to the size pricing rule. To implement this rule, managers can create a set $\text{Unit}_{s,i} \equiv \{(p_{s,i,l_1}, p_{s,i,k_2}): e_{s,i,k_1} p_{s,i,k_1}$ must cost less than $e_{s,i,k_2} p_{s,i,k_2}\}$, where $e_{s,t,k}$ is the multiplicative factor to convert equivalent units into whole product units. This leads to the following constraints:

$$\forall s,i \text{ and } (p_{s,t,k_1}, p_{s,t,k_2}) \in Unit_{s,i}\ e_{s,i,k_1} p_{s,t,k_1} \le e_{s,i,k_2} p_{s,t,k_2}$$

5. Line Pricing Rules:

Retail customers expect common prices for certain groups of products such as, for example, cough lozenges of the same brand but different flavors. These rules may be classified as line pricing rules and implement them as follows. Product groups called line price groups may be defined as $L_l, l = 1,..\|L\|$, where every product within a line group must have the same price in a store; i.e.,

$\forall L_l, l = 1,...\|L\|$, and $\forall k_1, k_2 \in L_l \; P_{s,t,k_1} = P_{s,t,k_2}$

6. Cluster pricing:

Retailers define geographic and other regions within which they maintain the same price for a given product. This may be translated to the notion of store clusters. A store cluster $Cluster_c$ is a set of stores such that the price of every product is invariant within the cluster. In other words, every product has the same price in every store within the cluster although each product may have a different price. In order to implement these constraints the set of stores may be partitioned into store clusters $Cluster_1,...,Cluster_{|c|}$ where $C \equiv \{\text{Cluster}_1,..., \text{Cluster}_{|C|}\}$, and a new price variable $P_{c,t,k}$ which represents the common price for product k in cluster $Cluster_c$ may be defined. This variable may be used in place of the original price variable $P_{s,t,k}$ whenever $s \in Cluster_c$.

Other rules may also be used. An example of such additional rules may be, but is not limited to, a gross margin rule, a store volume rule, or a competition rule.

A gross margin rule provides a constraint on the change of a mark up. A gross margin rule may provide a constraint on the change of the mark up of each individual item in a group or on the average mark up of the group or both. An example of a gross margin rule may be that the percentage change of the average gross margin of a group is not to exceed 5%.

A volume rule provides a constraint on the change or the volume of sales. The volume rule may provide a constraint on the change of volume of sales for each individual store or on the change of total volume of sales for a plurality of stores for an individual product or for a plurality of products. An example of a volume rule may be that for each store, the change of demand for a product is not to exceed 7%.

A competition rule provides a constraint on the difference between a competing store's prices and their prices. A competition rule may allow constraints for one or more competitors. An example of a multiple competition rule is that the price of each item must not exceed the price for the same item sold by competitor store X by more than 10% of the price and must not exceed the price of the same item sold by competitor store Y by more than 9% of the price. To implement these constraints a store may use monitors to check prices of items in competitor store X and competitor store Y.

The objective (such as profit) is then maximized. Such an objective would provide a maximized profit given the demand data and the constraints.

RULE RELAXATION

Given the objective and the plurality of constraints the solution may be infeasible due to a conflict between the objective and the plurality of

constraints. The objective may be expressed as an equation. Each constraint may also be expressed as an equation. A feasible solution may not exist that satisfies all equations representing the objective and the constraints. A preferred embodiment of the invention uses a rule relaxation process to provide a feasible solution when a feasible solution does not exit which satisfies all constraints and objectives.

During an optimization, it may be found that a rule is infeasible. A rule is determined to be infeasible, if either it cannot be satisfied when all other rules are satisfied or if an optimization cannot be performed without violating the rule. Although, one method of dealing with an infeasible rule is to drop the rule found to be infeasible, the preferred embodiment of the optimization step uses the rule relaxation process described below:

FIG. 6 is a flow chart of a preferred embodiment of the rule relaxation process. The rules are prioritized (step 604). A default prioritization may be provided, with an interface, which may allow a user to change the prioritization from the default. A check is made to see if a rule is infeasible (step 608). A rule is deemed to be infeasible if the relationship expressed by the rule is not able to be satisfied. For instance a rule $X+Y \leq 10$, is violated for when X is 7 and Y is 7 since then $X+Y=14$ which is greater then 10. If no rule is found to be infeasible, the rule relaxation process is stopped (step 628). If at least one rule is found to be infeasible, the lowest priority infeasible (LPI) rule is found (step 612). A determination is made whether rules with lower

priorities than the priority of the LPI rule may be relaxed to allow the LPI rule to become feasible (step 616). In the preferred embodiment the lower priority rules are checked before higher priority rules. If it is found that rules with lower priorities than that priority of the LPI rule may be relaxed to a point that allows the LPI rule to become feasible, then these rules with lower priorities are relaxed incrementally so that the LPI rule becomes feasible (step 620). In the preferred embodiment, lower priority rules are relaxed before higher priority rules. If it is found that rules with lower priorities than the priority of the LPI rule cannot be relaxed to allow the LPI rule to become feasible, then the LPI rule is relaxed until it becomes feasible (step 624). The rules are then rechecked to see if there are any remaining rules that are infeasible (step 608). This process is continued until all rules are feasible.

Example

To provide an example of rule relaxation, six categories of rules are provided. According to the method illustrated in the flow chart of FIG. 6, the categories of rules are prioritized (step 604) in the following order, with a ranking of 1 designating the category of rules with the highest priority and 6 designating the category of rules with the lowest priority.

1. Group price advance or decline rules. The user sets a maximum weighted group price advance or decline to 10%.

2. Size pricing rules. The user goes with the default that larger items cost less per equivalent unit than smaller identical items.

3. Brand pricing rules. For soft drinks, the user designates that the price of Brand A is never less than the price of Brand B. For juices the user designates that Brand C is always greater than Brand D.

4. Unit pricing rules. The user goes with the default that the overall price of larger items is greater than the overall price of smaller identical items.

5. Competition rules. The user designates that all prices must be at least 10% less than the prices of the same items sold by competitor X and are within 2% of the prices of the same items sold by competitor Y.

6. Line price rules. The user designates that different flavors of the same item are priced the same.

In this example, Brand A is sold by competitor Y for $8, while Brand B is sold by competitor Y for $10. In addition competitor Y sells one flavor of item C for $9 and another flavor of the same item C for $10.

A determination is made if a rule is infeasible (step 608). It is found that one of the price branding rules is infeasible, since Brand A is supposed to always have a price greater than the price of Brand B and yet competitor Y sells Brand A for $8 and Brand B for $10, and since the competition rules

designate that the same items must be within 2% of the prices of competitor Y. In addition, it is found that one of the line price rules is infeasible. This is because the line price rule constrains different flavors of the same item to be the same, but competitor Y is selling one flavor for $9 and another flavor for $10.

Since the branding rules have a priority of 3 and the line price rules have a priority of 6, it is found the line price rules are the lowest priority infeasible (LPI) rules (step 612). A check is made to see if rules with a priority lower than 6 may be relaxed so that the line price rules become feasible (step 616). Since the line price rules are the lowest priority rules, no rule with a lower priority may be relaxed so that the line price rule becomes feasible. As a result, the line price rule is relaxed allowing the different flavors to have different prices by about 10% (step 624).

The process then returns to step 608 to see if any rules are still infeasible. It is found that one of the price branding rules is infeasible, since Brand A is supposed to always have a price greater than the price of Brand B and yet competitor Y sells Brand A for $8 and Brand B for $10, and since the competition rules designate that the same items must be within 2% of the prices of competitor Y. Since the price branding rules are the only rules found to be infeasible, they are found to be the lowest priority infeasible (LPI) rules with a priority of 3 (step 612). A check is made to see if rules with priorities lower than the LPI rules may be relaxed to allow the LPI to be feasible (step

616). Preferably, the lower priority rules are checked first. So first the line price rules with a priority of 6 are checked to see if they may be relaxed to allow the branding price rules to be feasible. If not, then the next lowest priority rules, the competition rules are checked to see if they may be relaxed to allow the branding rules to be feasible. In this example, it is found by relaxing a competition rule with a priority of 5 the price branding rule becomes feasible. Therefore the competition rule is relaxed so that prices are within about 20% of the prices of competitor Y, which allows Brand A and Brand B to have the same price (step 620).

The process again returns to step 608 to see if any rules are still infeasible. It is found that all rules are now feasible, and the rule relaxation process is stopped (step 628).

In the preferred implementations, the specific infeasible solution at which the optimization process stops will depend on the optimization algorithm and the initial starting point. This means that specific rules that are relaxed in order to get to feasibility depend on both the optimization algorithm and the starting values These examples are merely illustrative and are not meant to limit the scope of the invention. In addition, rule priority was assigned by rule category. In other embodiments the rule priority may be assigned to individual rules instead of to rule categories.

Other specific methods of rule relaxation may be used. In general, rule relaxation methods provide for the prioritization of rules and cause lower priority rules to be relaxed before relaxing higher priority rules in order to make all resulting rules feasible. Another specific embodiment of this general rule relaxation method may allow different combinations of lower priority rules to be relaxed. These lower priority rules may be relaxed at the same rate or at different rates until the higher priority rule becomes feasible. Different combinations of the relaxation of lower priority rules may be specified in different embodiments.

Once the optimization (step 701) is completed, the prices of the products are sent to the stores. The prices of the products are then set to the optimized prices that are provided by the optimization.

SUBSET OPTIMIZATION

After an optimization, such as the optimization described above has been performed, new data may then be provided to the system (step 703). Such data may be provided through the Econometric Engine 104 (FIG. 1) or the Financial Model Engine 108, or through another source. If the entire product category was optimized using the new data, it is possible that all items in the product category would change because of the new data. To avoid an optimization that may cause a repricing of all products in the product category (since the cost of repricing is not negligible), the user is allowed to specify

subset limits (step 705). An example of such subset limits would be a maximum number N of products that may be repriced. A new optimization is performed using the new data, where the optimization is constrained to comply with the subset limits (step 707). For example, if a maximum number N of products may be repriced, the optimization provides an optimization with price changes of no greater than N products. Since the price of no more than N products is changed, instead of possibly the price of all products of the category, only a subset of the product category is optimized. The subset optimization may use the results of the previous optimization as initial values.

The subset optimization may choose the products that comprise this subset in a way that has the largest impact on the client's objective function. If, for example, the client's objective is to maximize profit, it is desirable to populate the subset of products whose prices are allowed to change with those products that are most likely to have the largest impact on profit. In one way of doing this, all of the available products may be ranked by their marginal contribution to the objective and then the subset of products whose prices are allowed to change is selected via mixed integer program.. The prices of all of the products not in the "optimal" subset may be frozen to provide a variant of the original scenario to obtain a new set of optimized prices.

A restatement of the process in FIG. 7, is as follows: The optimization of a product category may performed as discussed in the previous sections (step 701) to provide optimal prices, demand functions and constraints for all

of the products considered in the product category. All of the input data used in the initial optimization of the product category and $\vec{p}^{*}_{Cluster_z}$, the set of optimal prices derived from that data, may be used to provide initial conditions and parameters for the subset optimization.

Changes to the state of information that the user wishes to incorporate into the new optimization are also provided (step 703). For changes in costs, base price, or competitive prices, the change in the data from the original scenario measured as a percentage of base price is used. For discontinued or new products, the input data required to construct their demand functions and constraints as they would have been defined in the original optimization, as well as an initial starting base price, are used.

The subset limits, such as the maximum number of products that the user wishes to allow in the subset optimization, are specified (step 705). One possible assumption for an algorithm may be that the maximum number is at least as large as the number of products with information changes. A new optimization is then performed within the subset limits (step 707).

FIG. 8 is a more detailed flow chart of an embodiment of the step of optimizing product category within subset limits (step 707). A first step constructs a set of products that will be considered as candidates for the subset to be optimized (step 801). A second step chooses which of the candidates will be members of the subset (step 803), for example, by solving a mixed integer problem. A third step finds the new optimal prices for the products in the

subset while holding the prices of the other goods at their prior optimal level (step 805).

Construction of the Set of Candidates for Subset Optimization

The purpose of this step is to define a set of products, *C*, which will be candidates for subset optimization (step 805). In defining this set, priority is given to two distinct types of products: those products that have had changes in their state of information, and those products that have constraints that would allow their price to move if they are selected to be members of *C*.

Products with changes in state information are automatically included in *C* and in the subset of products to be optimized. The second group of products is chosen based on characteristics of their constraints and the direction of the gradient of the price variables with respect to the objective function at the incumbent optimal price. It is desirable to only choose products whose prices will actually be able to move in the subset optimization. If a product's price is at a bound in the original scenario and the gradient of the price variable indicates that the price will continue in the direction of the bound if the bound is relaxed, then the price will not move in the subset optimization since it will still hit that bound. If, however, the product's price was initially not at a bound or if it is at a bound but the gradient indicates that the price would move away from the bound if it were relaxed, then it would be expected that this price would change, if it is included in the subset

optimization. In order to discover the direction of the gradient of prices at their previous optimal price, a simple model is formulated that evaluates the original objective function while allowing prices to vary slightly from their previous optimal level, $\vec{p}^{*}_{Cluster_c}$. This model consists of the fundamental model equations that define demand group sales, market share, the user-defined objective function, and simple bounds holding prices within a small neighborhood of their previous optimal values. For a profit-maximizing problem, the model might be formulated as:

$$\text{Max} \sum_{i \in G} \sum_{k \in Dem_i} \hat{D}_{s,i,k,t} (P_{s,i,k,t} - C_{s,i,k,t})$$

$$\vec{P}^{*}_{Cluster_c} - \delta \le P_{s,i,k} \le \vec{P}^{*}_{Cluster_c} + \delta, \qquad \forall s, i, k$$

where δ is a small number.

λ^{u}_{k} and λ^{l}_{k} are defined as the marginal values on the upper and lower bounds defined in the final inequality constraints. These marginal values will be used as a measure of the price gradient as the price moves away from the incumbent optimal price. They, in concert with information about which bounds were active in the original problem, will determine which products are allowed to have price changes in the subset optimization.

Products for the second group should have the freedom to change prices during the subset optimization. To this end, all of the constraints in the initial model are evaluated and three operations are performed. First, a check is performed to see if that constraint was tight for the optimal price in the

original scenario. Essentially, this just means that those store-product combinations where the right and left-hand sides of the constraint equations are the same are found. For example, suppose that there is an upper bound on optimal price:

$$P_{c,i,k} \leq PMAX_{c,i,k}, \forall i,k$$

where c is the store cluster, i is the demand group, and k is the product. For this rule, all of the store-product combinations where

$$P_{c,i,k} = PMAX_{c,i,k}, \forall i,k$$

would be found.

Rule relaxation, discussed above, may have been utilized in the original optimization. As discussed above, rule relaxation allows the optimization to automatically relax constraints subject to a hierarchy defined by the user. Since the subset optimization is based on the original constraints of the initial scenario, it is possible to have optimal prices from the initial scenario that are infeasible with respect to some of the original constraints because those constraints were relaxed in the first scenario optimization. The second operation relaxes the bound on any store-product combinations for which the incoming optimal price is infeasible. Thus, for an upper price bound, we would set $PMAX_{c,i,k}$ equal to $P_{c,i,k}$.

The third procedure changes the initial bounds on those products which have had cost, base price or competitive price changes. If the cost, base price, or competitive price has increased, it is desirable to increase the upper bound

constraints so that the optimal price is allowed to move. The upper price bound, for example, would become

$$P_{c,t,k} \leq PMAX_{c,t,k}\ (1+\xi_{c,k}), \forall i,k$$

where $\xi_{c,k}$ is percentage change in cost or price with respect to base price. If the cost has increased, it is also desirable to raise the lower bound to prevent the price from flipping from the upper to the lower bound. The corresponding lower price bound would be

$$P_{c,t,k} \geq Max\left(PMAX_{c,t,k}\ (1-\hat{\lambda}_{c,k}), \vec{P}^{*}_{Cluster_c}\right) \forall i,k$$

where $\hat{\lambda}_{c,k}$ is a scaling factor for the allowable price region and $\vec{P}^{*}_{Cluster_c}$ is the optimal price for that product from the original scenario. If the cost or price has declined, then the lower price bound would become

$$P_{c,t,k} \geq PMIN_{c,t,k}\ (1-\xi_{c,k}), \forall i,k$$

and the upper bound becomes

$$P_{c,t,k} \geq Min\left(PMIN_{c,t,k}\ (1+\hat{\lambda}_{c,k}), \vec{P}^{*}_{Cluster_c}\right) \forall i,k$$

For constraints on the average price of a demand group, the bound is changed by the market share weighted average change in the cost or price relative to initial base price.

The set of candidates for subset optimization, C, may now be constructed as follows:

For each store, product and constraint:

1. Initialize C to include all available products.
2. If $\lambda_k^l > 0$ or $\lambda_k^u < 0$, then $k \notin C$

3. If constraint is an active upper bound on price and $\lambda_k^u > 0$, then $k \notin C$
4. If constraint is an active lower bound on price and $\lambda_k^l < 0$, then $k \notin C$
5. If product has change in the state of information, then $k \in C$

Selection of the Set of Products for Subset Optimization

In selecting the N products that will comprise the set of products, *R*, which are allowed to have price changes in the subset optimization, a simple mixed integer problem is solved. A binary variable, r_k, is created that indicates whether or not a given product is chosen to become a member of *R*. An objective function is the sum of these binary variables weighted by the maximum marginal value on each product's price obtained in the step of constructing the set of candidates for subset optimization. The only constraint is that the number of products chosen must be less than *n*, the user-defined maximum number of products allowed in the subset optimization minus the number of products with changes in the state of information. This problem can be written as

$$\underset{r}{Max}\left[\sum_{k \in C} r_k \cdot Max\left(\left|\lambda_k^l\right|, \left|\lambda_k^u\right|\right)\right]$$

subject to

$$\sum_{k \in C} r_k \leq n .$$

The set of products, *R*, may be defined to be considered in subset optimization as $k \in R : \{(k \in C \mid r_k = 1) \cup (k \in C \mid k \text{ has change in state})\}$.

Optimize Prices over the Product Subset R

The price variables of products that are not members of R are fixed and this new problem is solved with a complete optimization, such as the initial optimization described above, except that prices not in the subset are held fixed. Regression models and the predictive equations from the econometrics models are used to construct the optimization model. For example, the objective may be to maximize profit:

$$\sum_{i \in Gk \in Dem_i} \hat{D}_{s,i,k,t}(P_{s,i,k,t} - C_{s,i,k,t})$$

subject to constraining price changes to be within a given range of current prices.

$$PMIN_{s,i,k,t} \le P_{s,i,k,t} \le PMAX_{s,i,k,t}$$

To simplify notation, the time index may be removed from the equations. So the objective becomes:

Maximize: $\sum_{i \in Gk \in Dem_i} \hat{D}_{s,i,k}(P_{s,i,k} - C_{s,i,k})$.

When the subset optimization is completed and prices are optimized over the subset (step 805) a unique subset optimized data package is provided. The data package of the new prices for the subset of products is provided to the stores. The prices for the products in the subset, which is a smaller set

than all of the products, are changed (set) according to the optimized subset prices.

The subset optimization may be repeated several times between a complete optimization allowing all prices to change. For example, after subset optimization, new data may be provided. A user may specify new subset limits or use the previous subset limits. A new subset optimization may then be performed, using the specified subset limits and holding any frozen prices to the prices provided by the previous subset optimization.

In addition to the constraints listed above, the preferred embodiment may model several other business rules via constraints. These include limits on group price advance or decline, brand pricing rules, size pricing rules, and line pricing rules, as discussed above.

FIG'S. 9A and 9B illustrate a computer system 900, which is suitable for implementing embodiments of the present invention. FIG. 9A shows one possible physical form of the computer system. Of course, the computer system may have many physical forms ranging from an integrated circuit, a printed circuit board, and a small handheld device up to a huge super computer. Computer system 900 includes a monitor 902, a display 904, a housing 906, a disk drive 908, a keyboard 910, and a mouse 912. Disk 914 is a computer-readable medium used to transfer data to and from computer system 900.

FIG. 9B is an example of a block diagram for computer system 900. Attached to system bus 920 are a wide variety of subsystems. Processor(s) 922 (also referred to as central processing units, or CPUs) are coupled to storage devices, including memory 924. Memory 924 includes random access memory (RAM) and read-only memory (ROM). As is well known in the art, ROM acts to transfer data and instructions uni-directionally to the CPU and RAM is used typically to transfer data and instructions in a bi-directional manner. Both of these types of memories may include any suitable of the computer-readable media described below. A fixed disk 926 is also coupled bi-directionally to CPU 922; it provides additional data storage capacity and may also include any of the computer-readable media described below. Fixed disk 926 may be used to store programs, data, and the like and is typically a secondary storage medium (such as a hard disk) that is slower than primary storage. It will be appreciated that the information retained within fixed disk 926 may, in appropriate cases, be incorporated in standard fashion as virtual memory in memory 924. Removable disk 914 may take the form of any of the computer-readable media described below.

CPU 922 is also coupled to a variety of input/output devices, such as display 904, keyboard 910, mouse 912 and speakers 930. In general, an input/output device may be any of: video displays, track balls, mice, keyboards, microphones, touch-sensitive displays, transducer card readers, magnetic or paper tape readers, tablets, styluses, voice or handwriting

recognizers, biometrics readers, or other computers. CPU 922 optionally may be coupled to another computer or telecommunications network using network interface 940. With such a network interface, it is contemplated that the CPU might receive information from the network, or might output information to the network in the course of performing the above-described method steps. Furthermore, method embodiments of the present invention may execute solely upon CPU 922 or may execute over a network such as the Internet in conjunction with a remote CPU that shares a portion of the processing.

In addition, embodiments of the present invention further relate to computer storage products with a computer-readable medium that have computer code thereon for performing various computer-implemented operations. The media and computer code may be those specially designed and constructed for the purposes of the present invention, or they may be of the kind well known and available to those having skill in the computer software arts. Examples of computer-readable media include, but are not limited to: magnetic media such as hard disks, floppy disks, and magnetic tape; optical media such as CD-ROMs and holographic devices; magneto-optical media such as floptical disks; and hardware devices that are specially configured to store and execute program code, such as application-specific integrated circuits (ASICs), programmable logic devices (PLDs) and ROM and RAM devices. Examples of computer code include machine code, such as produced by a compiler, and files containing higher level code that are executed by a

computer using an interpreter. Computer readable media may also be computer code transmitted by a computer data signal embodied in a carrier wave and representing a sequence of instructions that are executable by a processor.

FIG. 10 is a schematic illustration of an embodiment of the invention that functions over a computer network 800. The network 800 may be a local area network (LAN) or a wide area network (WAN). An example of a LAN is a private network used by a mid-sized company with a building complex. Publicly accessible WANs include the Internet, cellular telephone network, satellite systems and plain-old-telephone systems (POTS). Examples of private WANs include those used by multi-national corporations for their internal information system needs. The network 800 may also be a combination of private and/or public LANs and/or WANs. In such an embodiment the price optimizing system 100 is connected to the network 800. Computer systems used by the stores 124 are also connected to the network 800. The computer systems for the stores 124 are able to bi-directionally communicate with the price optimizing system 100 over the network 800.

While this invention has been described in terms of several preferred embodiments, there are alterations, modifications, permutations, and substitute equivalents, which fall within the scope of this invention. It should also be noted that there are many alternative ways of implementing the methods and apparatuses of the present invention. It is therefore intended that the following

appended claims be interpreted as including all such alterations, modifications, permutations, and substitute equivalents as fall within the true spirit and scope of the present invention.